Facultatea Calculatoare, Informatica si Microelectronica

Universitatea Tehnica a Moldovei

Medii Interactive de Dezvoltare a Produselor Soft

Lucrarea de laborator#1

Version Control Systems

Autor:
Scripnic Alexandru

lector asistent:
Irina Cojanu
lector superior:
Svetlana Cojocaru

Laboratory work #2

1 Scopul lucrarii de laborator

Studierea unui Version Control Systems și lucrul cu versiunea CLI a acestuia. Examinarea instrucțiunilor de bază și lucru cu un repositoriu remote.

2 Obiective

– Intelegerea si folosirea CLI (basic level)

– Administrarea remote a masinilor linux machine folosind SSH (remote code editing)

– Version Control Systems (git —— bitbucket —— mercurial —— svn)

3 Laboratory work implementation

3.1 Tasks and Points

- initializeaza un nou repositoriu
- configureaza-ti VCS
- crearea branch-urilor (creeaza cel putin 2 branches)
- commit pe ambele branch-uri (cel putin 1 commit per branch)
- seteaza un branch to track a remote origin pe care vei putea sa faci push (ex. Github, Bitbucket or custom server)
- reseteaza un branch la commit-ul anterior
- folosirea fisierului .gitignore
- merge 2 branches
- rezolvarea conflictelor a 2 branches
- folosirea tag-urilor pentru marcarea schimbarilor simnificative precum release-ul.

3.2 Analiza lucrarii de laborator

Repository link

Primul pas a fost crearea unui nou repositoriu prin intermediul UI-ului oferit de github, iar mai precis apasând pe butonul New de pe pagina utilizatorului, tab-ul cu denumirea Repositories. După setarea numelui pentru repositoriu, am apăsat create repository.

Al doilea pas a constituit în crearea unui ssh key și l-am copiat în lista mea de key in account-ul github, ce mi-a permis editarea datelor din repositoriul creat anterior [1]. Apoi am clonat repositoriul local prin intermediul la instrucțiunea: git clone git@github.com:AScripnic/MIDPS-laboratories.git .

Mi-am configurat config-ul la git cli prin intermediul la intructiunile git config user.name "ascripnic" și git config user.email "scripnic.alexandru@gmail.com" [2].

În al treilea pas am create 2 branchuri și câte un commit în fiecare (referință: 3.1)prin intermediul comenzii git branch și denumirea branch-ului de care am avut nevoie și git commit -am și mesajul commitului. Este și posibilitatea de a crea și a te muta pe branchul create printr-o commandă git checkout -b. Cu ajutorul comenzii git push origin și denumirea la branch am încărcat schimbarile pe remote origin.

În al patrulea pas am resetat un branch, după ce am adăugat un fișier nou și i-am făcut commit,la commit-ul anterior (referință: 3.2) cu ajutorul comenzii git reset HEAD~ [3].

În al cincilea pas am adăugat un fișier în .gitignore (referință: 3.3) ce a făcut comenzile git să nu îl considere ca un fișier din proiect, cu excepția unor comenzi ce forțează git cli să lucreze cu acest fișier.

În al șaselea pas am făcut merge la 2 branchuri (referință: 3.4) cu ajutorul comenzii git merge și denumirea la branch-ul cu care aș dori să fac merge [4].

În al șaselea pas am creat un conflict între 2 branchuri, schimbând același fișier apoi am făcut git pull unul din altul, ce a generat un conflict (referință: 3.5). Sunt mai multe metode de a fixa un conflict, unele din ele ar fi:

- Manual - selectarea manuală a informației care trebuie sa rămână, și ștergerea celei inutile sau care nu mai este necesară
- Cu ajutorul comenzii git checkout [–theirs—ours] ce ar preoretiza schimbările în dependență de alegerea făcută [5].
- Cu ajutorul unor tool-uri care sunt prezente în unele IDE

Eu am ales varianta manuală (referință: 3.7) și o prefer mai mult ca a 2-a variantă pentru a înlătura riscurile de pierdere de date. Apoi am făcut commit cu fixarea făcută (referință: 3.8).

Pentru ultimul pas am creat un tag nou (referință: 3.9) cu ajutorul butonului Draft a new release de pe pagina `https://github.com/AScripnic/MIDPS-laboratories/releases`.

3.3 Imagini

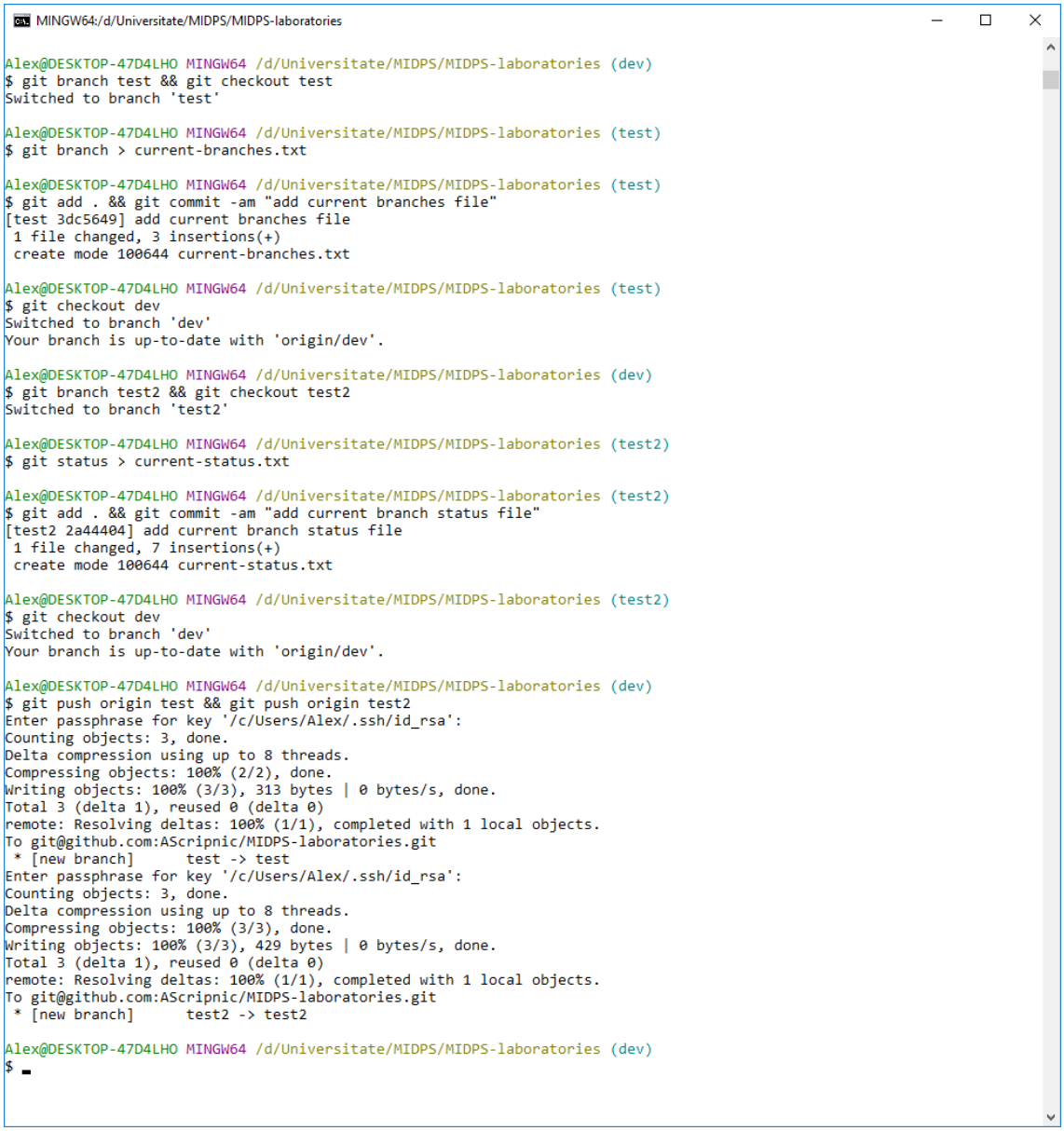

Figure 3.1 – Adaugarea a 2 branch-uri și câte un commit

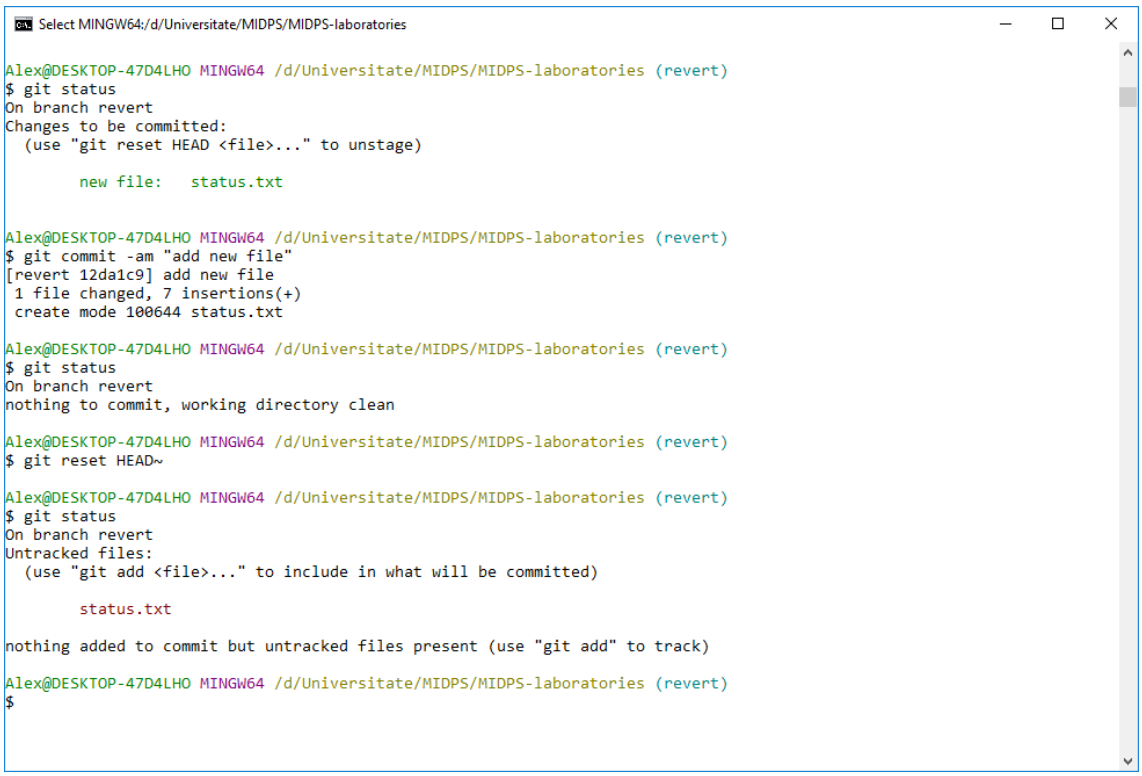

Figure 3.2 – Reset la commit anterior

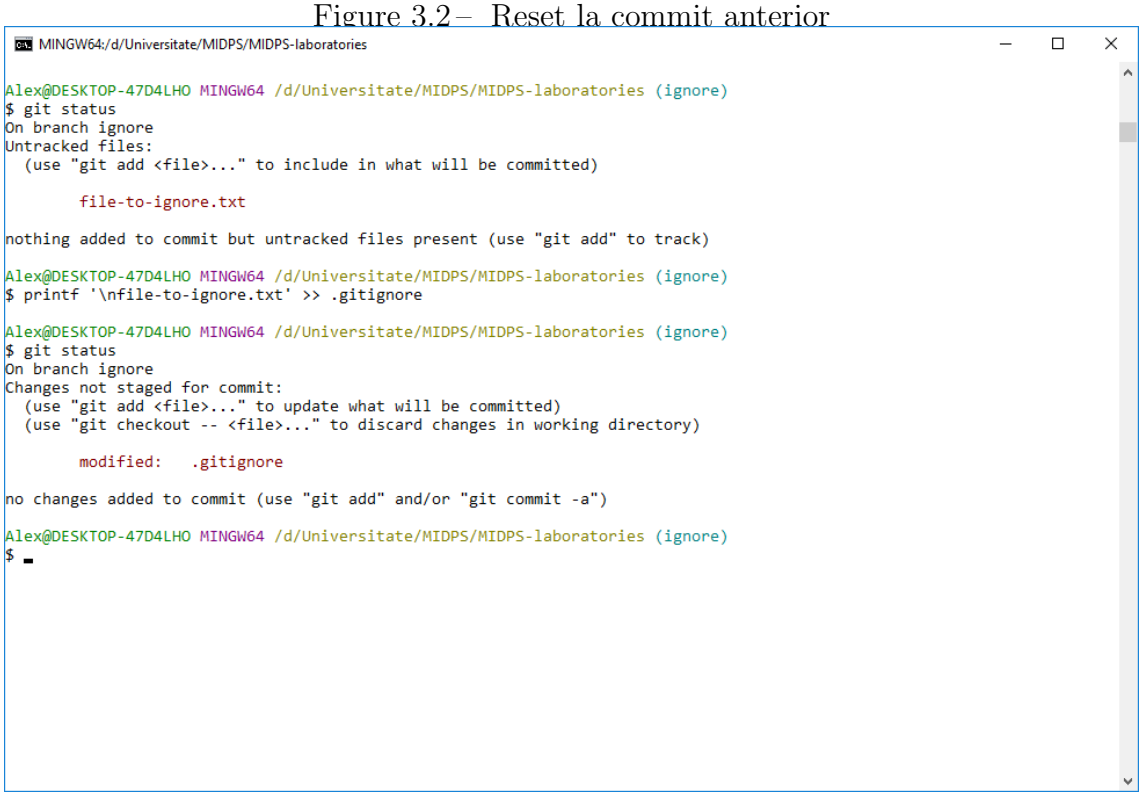

Figure 3.3 – Adăugarea unui fisier în .gitignore

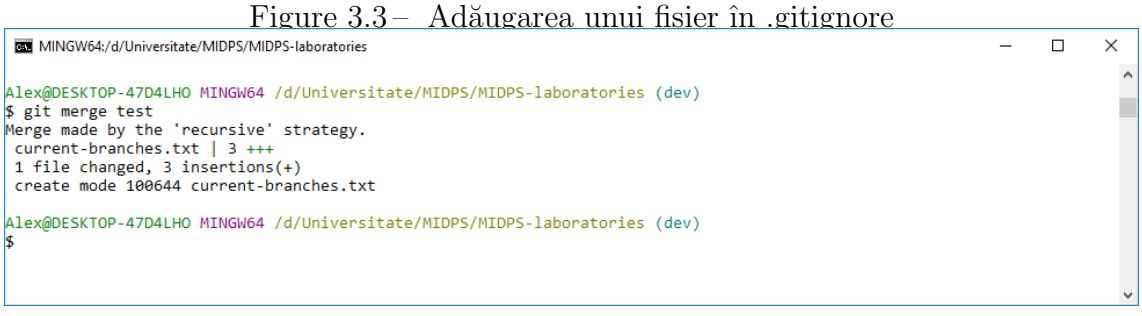

Figure 3.4 – Merge la 2 branch-uri

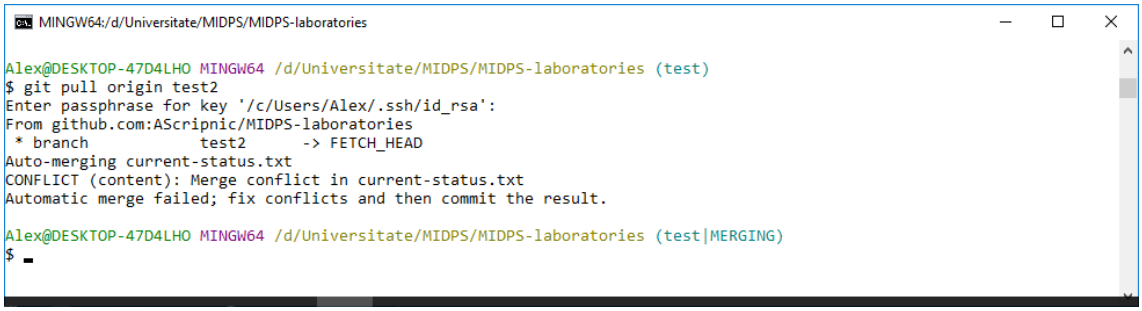

Figure 3.5 – Crearea unui conflict

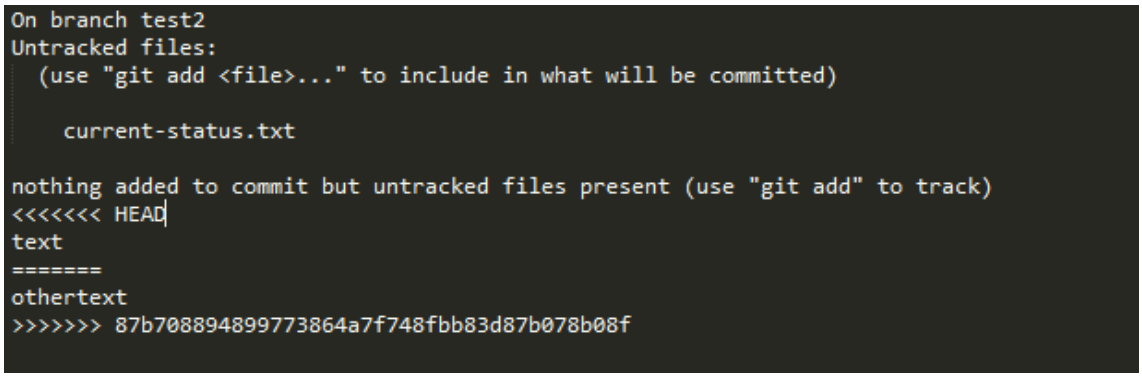

Figure 3.6 – Fișierul în care a avut loc conflictul

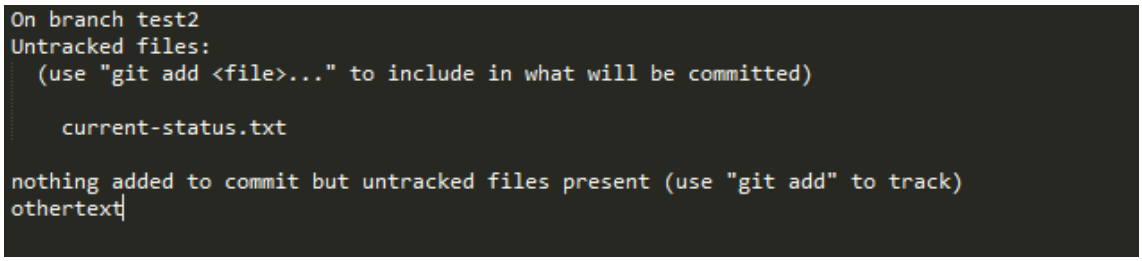

Figure 3.7 – Fișierul fixat după conflict

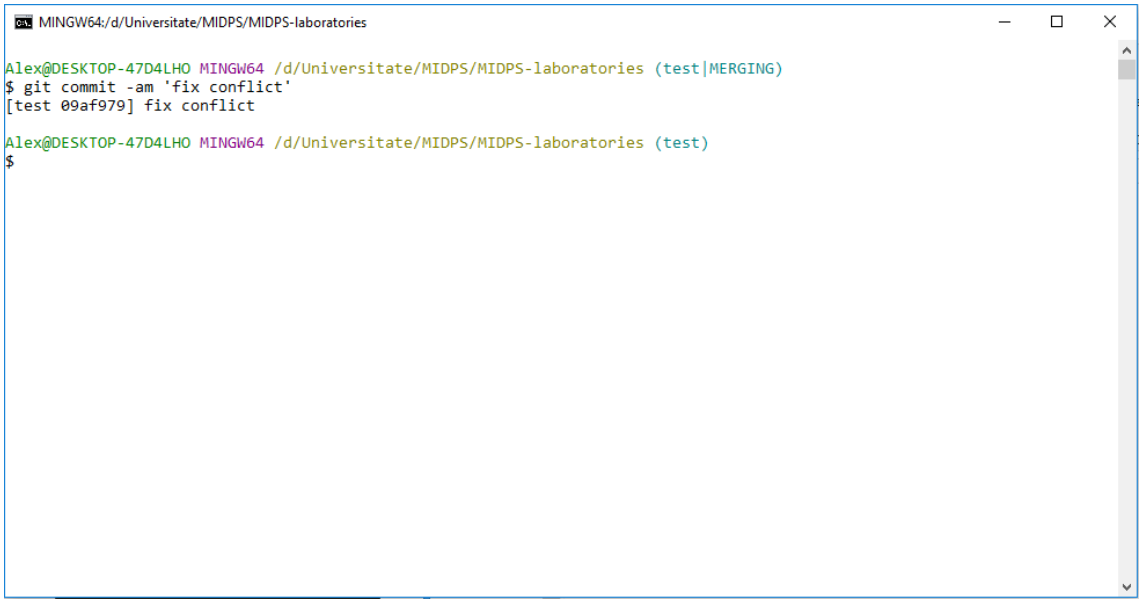

Figure 3.8 – Commit după rezolvarea conflictelor

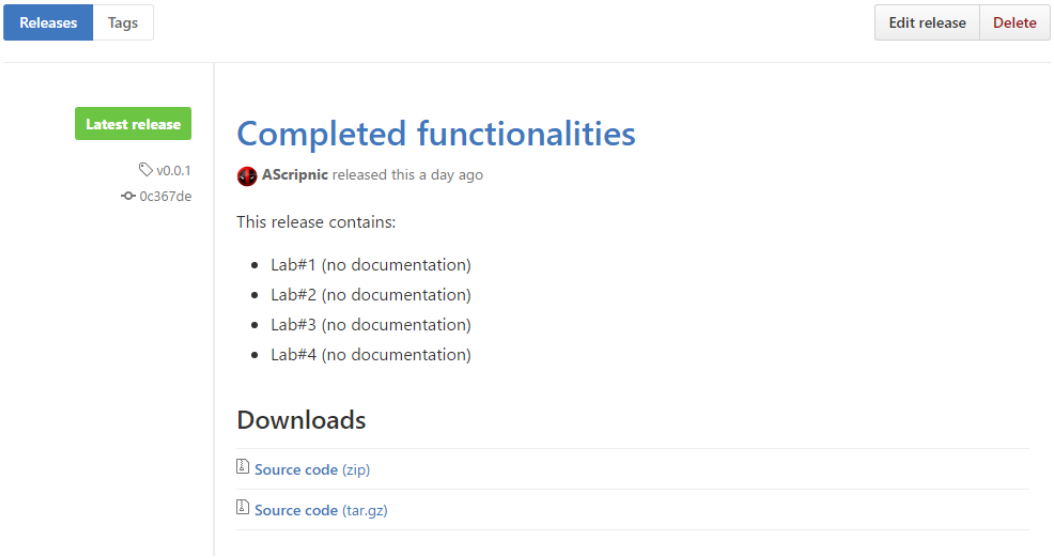

Figure 3.9– Crearea unui tag nou

Concluzie

Pe parcursul elaborării acestui laborator am studiat Version Control Systems, iar mai precis git și comenzile de bază în CLI.

Am descoperit noi metode de a face merge la un coflict între 2 branch-uri, cât și crearea, schimbarea, ștergerea unui branch. Am studiat posibilitatea de a face reset la un commit sau numai la un singur fișier.

References

1 Creating ssh key, *official page*, https://help.github.com/articles/generating-a-new-ssh-key-and-adding-it-to-the-ssh-agent/

2 Set git cli config, *official page*, https://help.github.com/articles/setting-your-username-in-git/

3 reset command, https://git-scm.com/docs/git-reset

4 merge command, https://git-scm.com/docs/git-merge

5 fix conflict via checkout command, http://gitready.com/advanced/2009/02/25/keep-either-file-in-merge-conflicts.html